ROME - Getting Started

May 1, 2001

Wolfgang Reißnegger
Distributed Systems Software Group
C&C Research Laboratories
NEC USA, Inc.
4 Independence Way
Princeton, NJ 08540-6634

You can get the current version of this document at http://rome.sourceforge.net
For questions and comments mailto:rome-admin@lists.sourceforge.net

Contents

List of Figures

1 Introduction

ROME is a modular, multitasking, embedded operating system which has been developed and used for multiple research projects within the Computer & Communications Research Laboratory (CCRL) of NEC USA, Inc. in Princeton, NJ.

ROME was designed to manage high speed data streams within a multimedia environment. The system is highly modular, with functionality split between multiple processes. To ensure a high throughput with minimal overhead ROME provides a *zero copy architecture* where pointer references to data are passed around instead of data being copied. The goal of this approach is to maximize the utilization of a given hardware configuration.

2 Purpose of this document

This document describes what kind of hardware and software is required in order to use the ROME tools and build your own ROME target system. It gives you a brief overview of what packages you need and where you can get them. It also provides some *basic* information of how to set up a ROME development environment.

If you want to get more detailed information about ROME and ROME software development, refer to the following documents:

- The ROME Programmer's Guide
- ROME Target Builder (RTB) User Manual
- The ROME Porting Guide
- ROME Modules/MessageSets/Targets Reference Pages

You can get all of these documents at `http://rome.sourceforge.net` in the *Documentation* section.

3 Requirements

ROME uses a cross-development platform. A target system is written, compiled and linked on a workstation or PC (host system) and the resulting image is transferred to the target hardware. No ROME "native" development tools are used, so you do not need to run a ROME system in order to write ROME code. There is (obviously) no requirement that the host architecture and the target architecture be the same, as long as a suitable cross-compiler is available.

3.1 Host system

The host system is where the editing, compilation and linking takes place. ROME development usually takes place on a Linux based host system. However, you can also work on any system running another

UNIX based OS (Sun Sparc, Mac PowerPC, Intel x86 PC) as long as the necessary tools are available. For example, RTB has been successfully compiled on a SunOS 2.6 machine.

If you can not use our pre-built toolkit, you will need a native C development environment (compiler, linker etc.) for your actual host system.

3.2 Host software

In order to run RTB you will need the following packages installed on your system:

- The *gtk+ toolkit, version 1.2.1 or higher*. If you are running a recent Linux distribution, then these libraries are likely to be already installed. Otherwise you can get them from `http://www.gtk.org/` and compile them for your system.
- *CVS, version 1.10 or later*, installed on your system. CVS is usually included in the default installation of most Linux distributions. Type `cvs -v` in a shell to find out whether or not CVS is installed and which version you have.

To compile the source code you will also need:

- The *gtk+ toolkit development* package, *version 1.2.1 or later* (which contains the header files, also available at `http://www.gtk.org/`);
- A C-compiler ;-)

RTB does not *strongly* rely on gtk+ version 1.2.1 or higher. You might be able to compile or run it with an older version of gtk+. However, older versions have not been tested.

4 Getting ROME

You can get all necessary ROME packages at `http://rome.sourceforge.net/`. This server contains the sources for ROME and for the ROME development tools.

4.1 Getting ROME via CVS

You can use SourceForge's CVS server to check out the sources and the documentation. If you are running Linux, enter the following commands in a shell:

```
export CVSROOT= \
   :pserver:anonymous@cvs.rome.sourceforge.net:/cvsroot/rome
cvs login
```

You do not have to enter a password. Now you can checkout the ROME repository using the command:

```
cvs -z3 checkout src
```

for the ROME sources (including RTB) and:

```
cvs -z3 checkout docs
```

for the document tree.

4.2 Downloading ROME .tar.gz packages

You can also download the sources in .tar.gz format. See the *Download* section on ROME homepage at http://rome.sourceforge.net. Sources in the .tar.gz packages are usually more stable than the *"leading edge"* CVS snapshots.

5 Development environment

Figure 1 shows an overview of a typical ROME development environment (excluding the target system). Usually there are three components involved:

- SourceForge Server;
- Hosts in the local network;
- Local CVS server.

SourceForge Server

The SourceForge Server is only needed to obtain the sources for ROME and the ROME tools. If you are not an active contributer to the ROME project you will not have to deal with this server very often.

Local Hosts

These are used to develop and cross-compile ROME systems. You will install and run RTB on these machines to configure and build your targets.

Local CVS server

This CVS server is an optional component in a ROME development environment. A CVS server brings a lot of advantages, as it allows you to track the revisions of your software and allows you to go back to a certain revision of a file. Even if you are the only developer in your network, a CVS server can make sense. In this case you are using CVS *locally*, which means that you keep all the sources in a local directory and let CVS manage them.

In a distributed development environment with multiple developers, a CVS server is highly recommended. In this case you need to run a CVS server on a shared machine (much like the SourceForge CVS server). Developers can log in and check in/out sources from the CVS server. If you have CVS installed on your system, you will likely find some documentation in /usr/doc/cvs-x.x.x/ where x.x.x is the version of your CVS package. Read the documentation to learn how to set up CVS on your machine for your needs.

If you really don't know how to do this, ask your system administrator (if you have one).

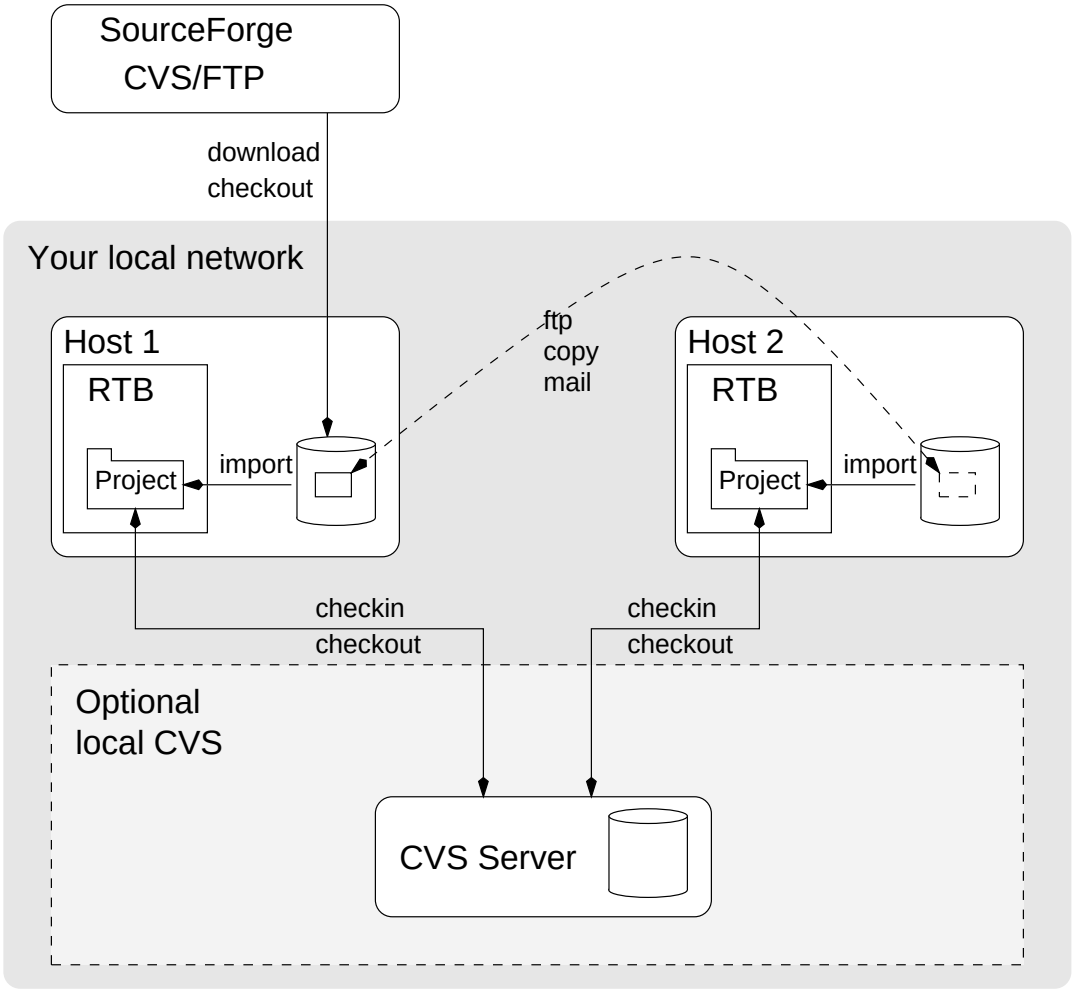

Figure 1: ROME development environment

6 Typical development setup

You can see a typical development setup in Figure 2. Usually, a PC based host machine is attached to the

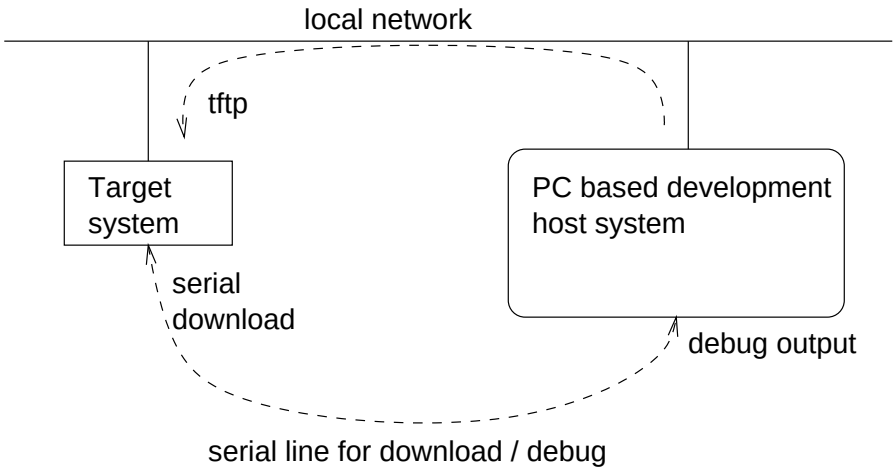

Figure 2: Typical ROME development setup

target system via a serial cable. If the target system can be connected to a local network it might make sense to boot the target file via tftp to speed things up a little.

7 Installing RTB

Installing RTB is relatively simple. You can either check out or download the sources from SourceForge CVS and compile it yourself, or you can download the pre-compiled binary from SourceForge and install it somewhere in your `$PATH`. To compile it yourself get the sources, change into the distribution directory (e.g. `rtb-0.7.5`), then type:

```
./configure [--prefix=/your/installation/path]
```

As you can see, you can provide your own installation path in case you don't have access to the default install directory (`/usr/local`). Check the output for any obvious errors (missing header files, libraries or outdated versions thereof). If everything looks OK type:

```
make
```

You will (hopefully) get a `rtb` executable in the `src` directory and some small utility programs in the `utils` directory. Now you can install the programs typing:

```
make install
```

If you have any problems you can also uninstall the programs typing:

```
make uninstall
```

If you want to use the pre-compiled version, download it from SourceForge and type (as root)

```
cd /
tar -xzvf /your/download/path/ROME-RTB-i386.tar.gz
```

The executable will be extracted into `/usr/local/bin` and is ready to run.

8 Cross compiler setup

In order to compile ROME for target architectures that are not the same as your host system, you have to install the cross compiler tools. The ROME distribution contains pre-built cross-compilers for the Intel I960 and the MIPS architectures, but you will probably have to make your own set of cross compiler tools. This chapter is not going into details of how to do this, because it would take to much space and actually does not really belong here. There is a very useful FAQ on how to build a cross compiler using the GNU gcc toolchain. You can find it at `http://www.objsw.com/CrossGCC` or at `ftp://ftp.objsw.com/pub/crossgcc`. Follow the instructions and build all the tools you need (`gcc`, `as`, `ld` etc.)

8.1 Cross compiler directories

Now you need to set up a directory where all the executables of the cross-compiler will reside so RTB can find them. This can be local directory in your `$HOME` or a system directory (e.g. `/usr/local/rtb/xgcc`). You will have to configure this directory in the RTB *Preferences* dialog as the *Tools directory*. RTB will then create the correct `PATH` settings for the `Makefiles` to compile a ROME target.

RTB also expects the cross compiler tools to reside at a certain place within the *Tools directory*. All tools for a certain architecture should be placed in a directory named as follows:

```
/tools_directory/CPU_Class/CPU_Type/
```

where `CPU_Class` must be set according to the *Class field* and `CPU_Type` according to the *Type field* in the Target settings for you project. For the provided P4032 Target these values are, for example, `MIPS` and `r4300`.

8.2 Non-gcc compilers

If you are using a compiler tool set which has not been cross compiled from the GNU gcc package, you might have to rename the names of the tools to confirm with RTB. RTB assumes the following naming:

C compiler	`gcc`
Assembler	`as`
Linker	`ld`

You can either rename your tools when you put them in the Tools directory or create symbolic links.